COMP9414 Artificial Intelligence

`UNSW` `COMP9414`

Outline 如果以下的内容中有出现错误，请联系159595223@qq.com 修改

祝大家考出好成绩

- **Course_name** ：COMP9414
- **Lecturer** ：Alan Blair
- **Author** ：Zeshi Wu

---

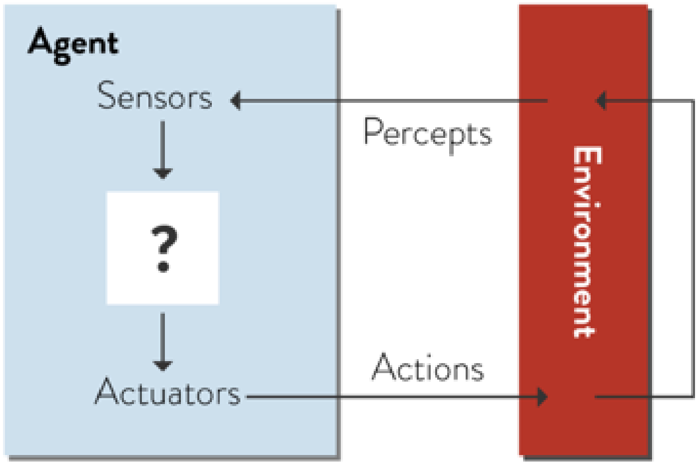
week1

week2 Classifying AI Tasks & Agent Type

2.1 Example of AI Tasks

- WK2: Wumpus World, Robotcup Soccer
- WK3: Path PLanning(迷宫)
- Wk4: Path Search Puzzles(魔方)
- WK5: Games (国际象棋，黑白棋，卡牌)
- WK6: Constrain Satisfaction(N-queens,数独)

2.2 Agent Model



2.3 Agents as Functions

- Agents 可以根据经验或数学公式来evaluated
- Agents 是一个连续感知的功能
- 理想的Agent 能自己选择预计实现最大性能的措施

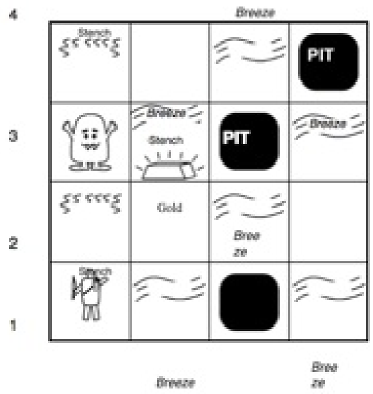
2.4 The PEAS model of an Agent

- Performance measure 任务完成 / 失败条件
- Environment 环境描述
- Actuators 执行器
- Sensors 感应器
  都是概念类的知识点，e.g. 象棋，taxi

2.5 Wumpus World

Environment

- Squares adjacent to Wumpus are Smelly
- Squares adjacent to Pit are Breezy
- Glitter iff Gold is in the same square
- Shoot
  - kills Wumpus if you are facing it
  - uses up the only arrow
- Grab
  - picks up Gold if in same square



Performance measure

- Return with Gold +1000, death -1000
- -1 per step, -10 for using the arrow

Actuators

- Left, Right, Forward, Grab, Shoot

Sensors

- Breeze, Glitter, Stench

2.6 Classifying Tasks

Simulated	vs.	Situated or Embodied
Static	vs.	Dynamic
Discrete	vs.	Continuous
Fully Observable	vs.	Partially Observable
Deterministic	vs.	Stochastic
Episodic	vs.	Sequential
Known	vs.	Unknown
Single-Agent	vs.	Multi-Agent

2.7 Environment Types

Simulated: a separate program is used to simulate an environment, feed percepts to agents, evaluate performance, etc.

Static: environment doesn't change while the agent is deliberating

Discrete: finite (or countable) number of possible percepts/actions

Fully Observable: percept contains all relevant information about the world

Deterministic: current state of world uniquely determines the next

Episodic: every action by the agent is evaluated independently

Known: the rules of the game, or physics/dynamics of the environment are known to the agent

Single-Agent: only one agent acting in the environment

2.8 看PPT 各类的对比例子（缺）

2.9 Agent Type

- Reactive Agent
- Model-Based Agent
- Planning Agent
- Game Playing Agent
- Learning Agent

2.9.1 Reactive Agent

- 仅根据当前感知到的下一个动作选择使用“策略”或一组简单适用的规则有时pajoratively称为“简单反射剂”–但他们可以做的令人惊讶的复杂的事情
- Limitations:
  - Reactive Agents have no memory or “state”
    - ◭ unable to base decision on previous observations
    - ◭ may repeat the same sequence of actions over and over
  - This phenomenon can also be observed in nature ◭ wasp dragging stung grasshopper into its nest

```
魏政：
最简单的代理，agent直接通过percept来确定下一步的action，就是观察到什么，就做什么
虽然简单，但是挺实用 e.g. Swiss robots, simulated hockey
```

```
缺点是没有memory，无法记住之前的percept，所以可能反复的执行同一个没意义的操作（因为没记住以前这个没意义的操作）
```

2.9.2 Model-Based Agent

- Advantages
  - Model-Based Agent can keep a “map” of the places it has visited, and remember what it perceived there.
- Limitation
  - Sometimes we may need to plan several steps into the future.

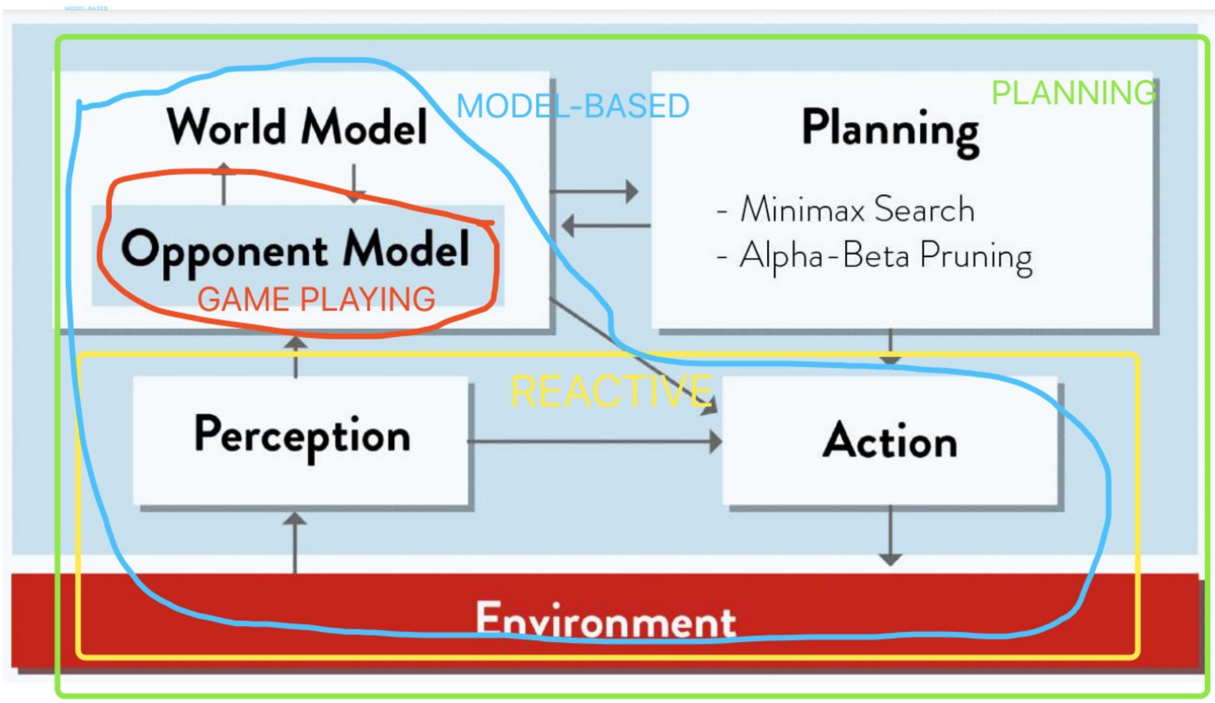
```
魏政：
在简单反射代理上引入了world model，agent可以记住自己之前访问的地图，所以避免了重复做一件没意义的action的情况
缺点是没有plan，所以agent只能查看过去，不能计划未来，如果做下面一些事的话就会表现的很糟糕
    1.有想下几步怎么做的任务，比如下棋和魔方
    2.有顺序的任务，比如做饭和修手表
    3.有目标的任务，比如旅行去某个城市
```

2.9.3 Planning Agent

- Sometimes an agent may appear to be planning ahead but is actually
- These rules can be hand-coded, or learned from experience.
- Agent may appear intelligent, but is not flexible in adapting to new situations.

```
魏政：
如图，在基于模型的代理上引入了planning
使得在world model里有了对应的目标，比如
    碰到了闪闪发光的东西就抓住
    碰到了臭气就射击
这些plan一般都是hardcode或者基于经验写死在程序里的，所以缺点就是agent表现出intelligent，但是不会自己去适应新环境
```

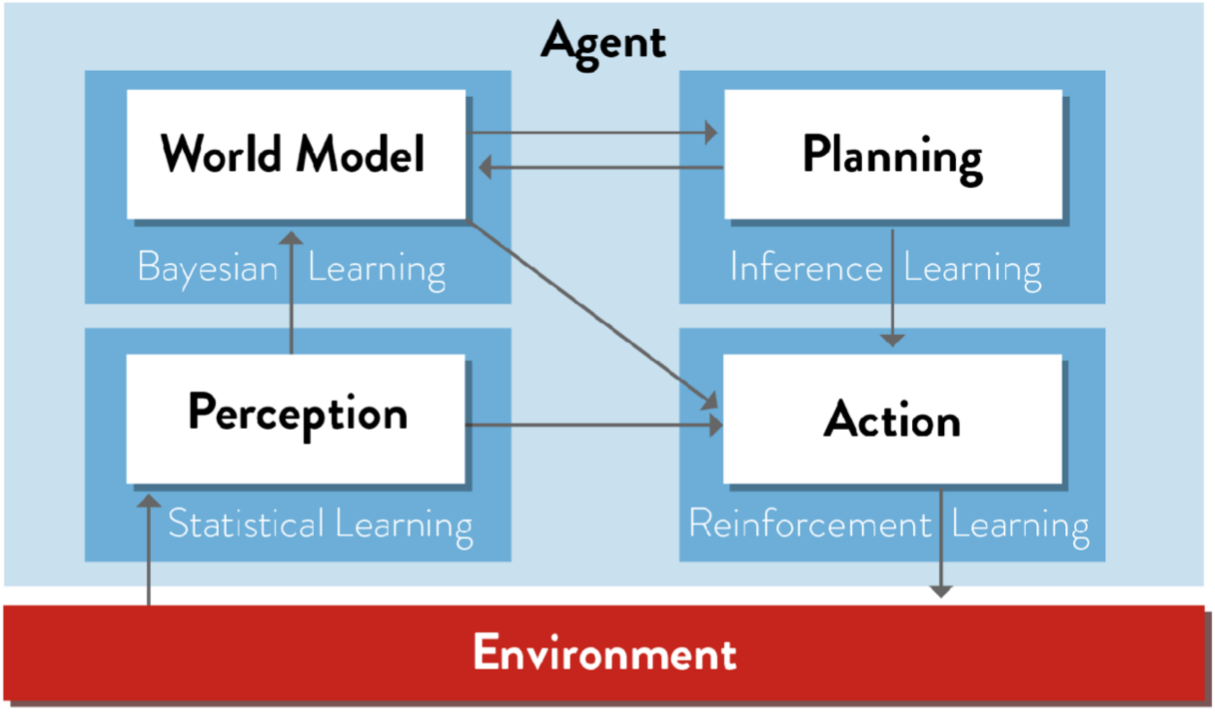
2.9.4 Game Playing Agent

- 


```
魏政：
在world model里加入了对手，可以多玩家一起玩
注意planning是针对多玩家的minimax search 和 alpha-beta pruning
```

2.9.5 Learning Agent

- Learning is not a **separate module**, but rather a set of techniques for improving the existing modules
- Learning is necessary because:
  - may be difficult or even impossible for a human to design all aspects of the system by hand
  - the agent may need to adapt to new situations without being re-programmed by a human

- 


```
魏政：
分别在perception，action，world model和planning里加入了学习机制
学习并不是一个单一的模块，而是为了改善这已经存在的四个模块
学习是有必要的，因为
    人类手工完成四个模块的设计有时候太难，甚至不可能，所以需要模块自主学习
    遇到了新环境，agent可以自适应，不需要每次都要人类来加入新的规则
我们要分清，学习复杂并不代表应用复杂，比如说，让一个agent学习曲棍球可能要很久（学习复杂），但是一旦学会之后，就可以立即拿来用（应用并不复杂）
```

week3 Path Search

3.1 Motivation

- Reactive和Model_Based Agnets根据目前及过去的获知来采取行动
- 两个搜索策略
  - Uninformed 盲搜策略只能将目标状态与非目标状态区分开来
  - INformed 知情搜索策略使用启发式方法来尝试“更接近”目标

3.2 罗马尼亚街道图

现在在Arad，明天去Bucharest，无法退票

A task is specified by states and actions:

- state space e.g. other cities
- initial state e.g. “at Arad”
- actions or operators (or successor function $S(x)$)
  e.g. Arad → Zerind Arad → Sibiu etc.
- goal test, check if a state is goal state
  In this case, there is only one goal specified (“at Bucharest”)
- path cost e.g. sum of distances, number of actions etc.

3.3 Search Tree (概念类无用)

- Seartch tree: 状态空间叠加
- Root: 根节点对应初始节点
- Leaf Nodes:叶节点：对应于树中没有后继的状态，因为它们没有展开或生成没有新节点
- 状态空间与搜索树不一样 *20state=20cities中的寻路为例，但有无限多的路径！*

3.3.1 Data Structure for a Node

One possibility is to have a node data structure with five components:

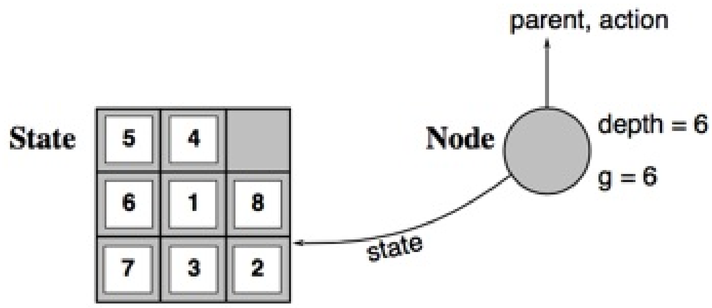
1. Corresponding state
2. Parent node: the node which generated the current node.
3. Operator that was applied to generate the current node.
4. Depth: number of nodes from the root to the current node.
5. Path cost.

3.3.2 States vs. Nodes

状态是（表示）物理配置

一个节点是一个数据结构组成的搜索树的一部分，包括父母，孩子，深度，路径成本G（X）

状态没有父母，孩子，深度，或路径成本！



3.3.3 Data Structures for Search Trees

Frontier：收集有待扩展的节点
它可以作为优先级队列执行以下操作：
make-queue（items）创建了项目队列。
如果队列中没有项目，布尔空（队列）返回true。
remove-front（队列）移除队列前的item并返回。
queueing-function（items、队列）插入新的items进入队列。

3.4 Search Stragegies

通过选择节点扩展顺序来定义策略
战略评估如下尺寸：

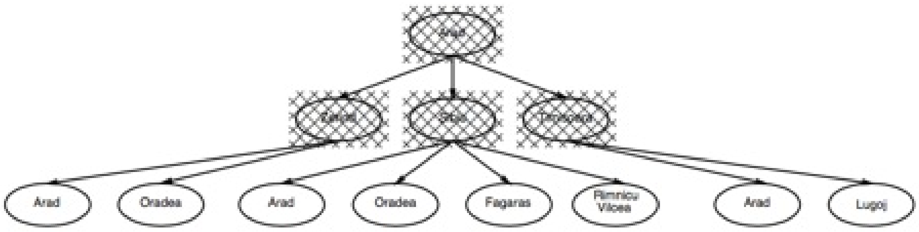
- 完整–它是否总找到一个解决方案，如果方案存在？
- 时间复杂度–生成/扩展节点的数量
- 空间复杂度–在内存中最大节点数
- 最优–它总是找到一个成本最低的解决方案？
- d: depth of 成本最低的解决方案
- b: 最大分支系数进行测量的时间和空间复杂度
- m: 最大深度的状态空间（可能是∞）

3.5 Uninformed search strategies (latex公示未补全&表格)

- Breadth First Search 广搜
- Uniform Cost Search
- Depth First Search 深搜
- Depth Limited Search

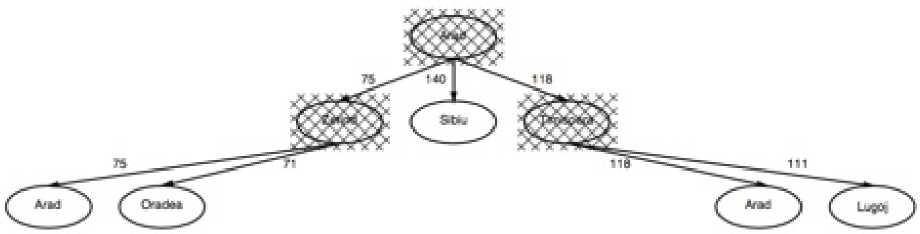
- Iterative Deepening Search

3.5.1 Breadth First Search

- implementation: QUEUEING-FUNCTION = put newly generated successors at end of queue
- Very systematic
- **Complete**: Yes (if b is finite the shallowest goal is at a fixed depth d and will be found before any deeper nodes are generated)
- **Time**: 1+b+b2 +b3 +...+bd = bd+1−1 = O(bd) b−1
- **Space**: O(bd) (keeps every node in memory; generate all nodes up to level d)
- **Optimal**: Yes, but only if all actions have the same cost
- 

3.5.2 Uniform-Cost Search

- 先找根节点，然后找成本最低的节点
- Implementation: QUEUEINGFUNCTION = insert nodes in order of increasing path cost.路径成本递增
- Reduces to Breadth First Search when all actions have same cost
- **Complete**: Yes,ifbisfiniteandstepcost≥εwithε>0
- **Time**: O(b⌈C∗/ε⌉) where C∗ = cost of optimal solution, and assume every action costs at least ε
- **Space**: O(b⌈C∗/ε⌉) (b⌈C∗/ε⌉ = bd if all step costs are equal)
- **Optimal**: Yes.



3.5.3 Depth First Search

- **Complete**? No! fails in infinite-depth spaces, spaces with loops; modify to avoid repeated states along path ⇒ complete in finite spaces
- **Time**: O(bm) (terrible if m is much larger than d but if solutions are dense, may be much faster than breadth-first)
- **Space**: O(bm), i.e. linear space!
- **Optimal**? No, can find suboptimal solutions first.

3.5.4 Depth Limited Search

- 如深度优先搜索扩展节点，但规定的最大路径深度的截止。
- **Complete**? Yes (no infinite loops anymore)
- **Time**: O(bk), where k is the depth limit
- **Space**: O(bk), i.e. linear space similar to DFS
- **Optimal**? No, can find suboptimal solutions first

3.5.5 Iterative Deepending Search

- Tries to combine the benefits of depth-first (low memory) and breadth-first (optimal and complete) by doing a series of depth- limited searches to depth 1, 2, 3, etc.
- **Complete**? Yes
- **Time**: O(bd)
- **Space**: O(bd)
- **Optimal**? Yes, if step costs are identical.

3.5.6 Complexity Results of Uniformed Search 背

Criterion	Breadth-First	Uniform-Cost	Depth-First	Depth-Limited	Iterative Deepening
Time	$O(b^d)$	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(b^m)$	$O(b^k)$	$O(b^d)$
Space	$O(b^d)$	$O(b^{\lceil C^*/\varepsilon \rceil})$	$O(bm)$	$O(bk)$	$O(bd)$
Complete?	Yes[1]	Yes[2]	No	No	Yes[1]
Optimal ?	Yes[3]	Yes	No	No	Yes[3]

b = branching factor, d = depth of the shallowest solution,
m = maximum depth of the search tree, k = depth limit.
1 = complete if b is finite.
2 = complete if b is finite and step costs $\geq \varepsilon$ with $\varepsilon > 0$.
3 = optimal if actions all have the same cost.

3.6 Informed(heuristic) search strategies

- more efficient than Uninformed

week4 Heuristic Path Search

✏ Goal: be able to recognize and use greedy and A* algorithm

4.1 Search Strategies

4.2 Informed Search & Heuristic 高频考点

- heuristic function h(n)
  - 估计函数$f(n) = g(n) + h(n)$
    - g(n) : path from the 初始点到 n
    - h(n) : 估计最小代价 from n to the 目标
  - $h(n) = 0 \to f(n) = g(n)$:UCS(e.g.Dijkstra),只需求出起点到任意顶点n的最短路径g(n)，而不计算任何评估函数h(n), 需要计算最多的定点
  - $g(n) = 0 \to f(n) = h(n)$: 贪心BFS, 只考虑当前点n到后面的距离尽可能 小，速度最快，但可能得不出最优解
  - $f(n) = g(n) + h(n)$: A*

4.3 Greed Best-First Search

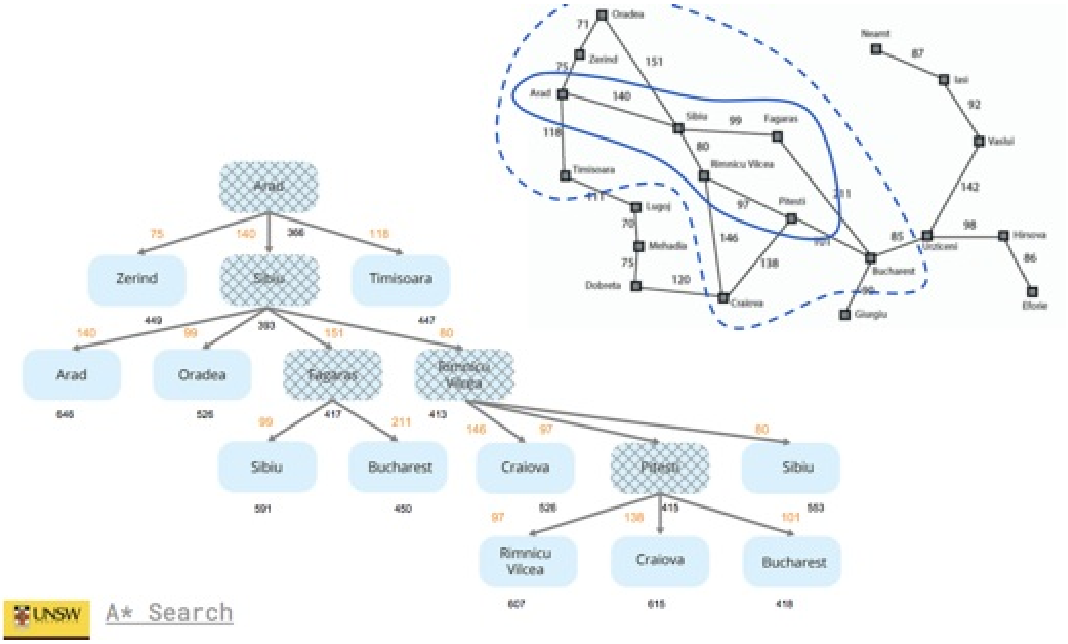
- Selects the next node for expansion using the heuristic function for its evaluation function
- Minimises the estimated cost to the goal expands whichever node n is estimated to be closest to the goal
- 仅考虑当前局部最佳, 一定程度上降低了复杂度, 但可能得不到全局最优

- **Complete**: No! can get stuck in loops, e.g.,Iasi → Neamt → Iasi → Neamt → ... Complete in finite space with repeated-state checking
- **Time**: O(bm), where m is the maximum depth in search space.
- **Space**: O(bm) (retains all nodes in memory)
- **Optimal**: No! e.g., the path Sibiu → Fagaras → Bucharest is 32 km longer than Sibiu → Rimnicu Vilcea → Pitesti → Bucharest.
- 因此贪婪搜索□深度优先搜索具有相同的问题。然而，一个好的启发式可以大大减少时间和内存成本。

4.4 Uniform Cost Search(见3.5.2)

4.5 A* Search

- Greedy minimize h(n) efficient but not optimal or complete
- UCS minimize g(n) optimal and complete but not eficient
- A* Search集合了以上贪婪和UCS的长处,保证了效率的同时避免扩展已经很expensive的路径

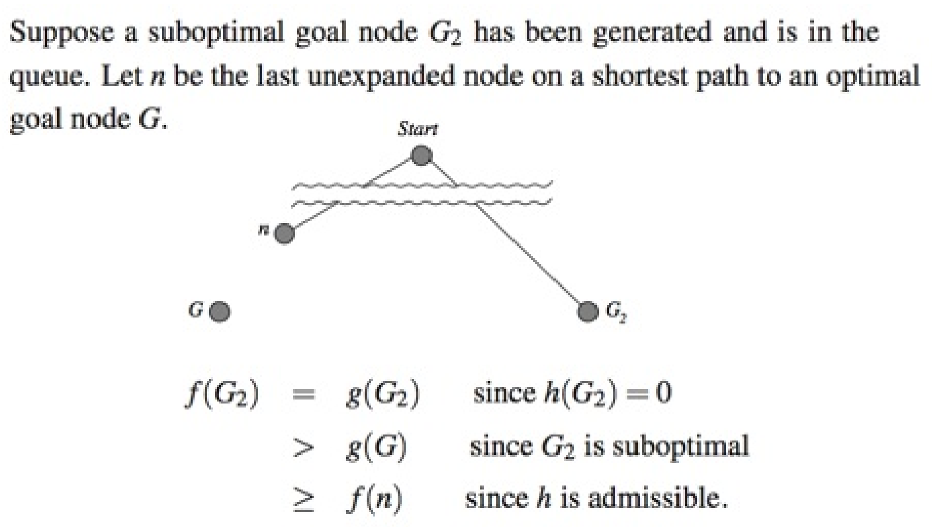


- Simulation:
  - 加粗体文字初始化为该点到Bucharest的直线距离(E.G. Arad = 356, Sibid = 253)
  - 接着点n的值被更新为出发点Arad经过点n后到Bucharest的直线距离(E.G. Sibiu = 140 + 253 = 393)

- 从Arad出发的三条路径中, 到Sibiu再前往Bucharest的距离最短, 因此接下来从Sibio出发 - 从Sibio出发的四条路径中, 到Rimnicu Vilcea再前往Bucharest的距离(413)最短, 因此接 下来从Rimnicu Vilcea出发
- 从Rimnicu Vilcea出发的三条路径中, 到Pitesti再前往Bucharest的距离(415)最短, 因此接 下来从Pitesti出发
- 从Pitesti出发可到达Bucharest, Bucharest的值被更新为418, 小于从Arad出发经Sibid,Pagaras的距离加上Pagaras到Bucharest的直线距离, 因此接下来从Pagaras出发
- 从Pagaras出发可到达Bucharest, 但Bucharest的值将被更新为450 > 418, 因此最终解仍 为: $A \to S \to R \to P \to B$
- Admissible of h(): 假若从n到终点的真实距离始终不小于h(n), 则该启发式函数是admissible的

4.6 Proof A* Search is Optimal

4.7 Optimality of A* Search



- A* Search是optimal and complete的 → h(n)不会过度估计到达目标的成本

week5 Games

week6 Constraint Satisfaction

魏政
https://docs.google.com/document/d/1x4LIVsiNy9A1e5I3E3VfQaswLWrFOrFCC1-qD5hCZBs/edit

week7 放假

week8 Logic agent

- comp9020

考点

- Constants : Gold,Wumpus,[1,2],[3,1], etc.
- Predicates : Ad jacent(),Smell(),Breeze(),At()
- Functions : Result()
- Variables : x, y, a, t,...
- Connectives : ∧ ∨ ¬ ⇒ ⇔
- Equality : =
- Quantifiers : ∀ ∃

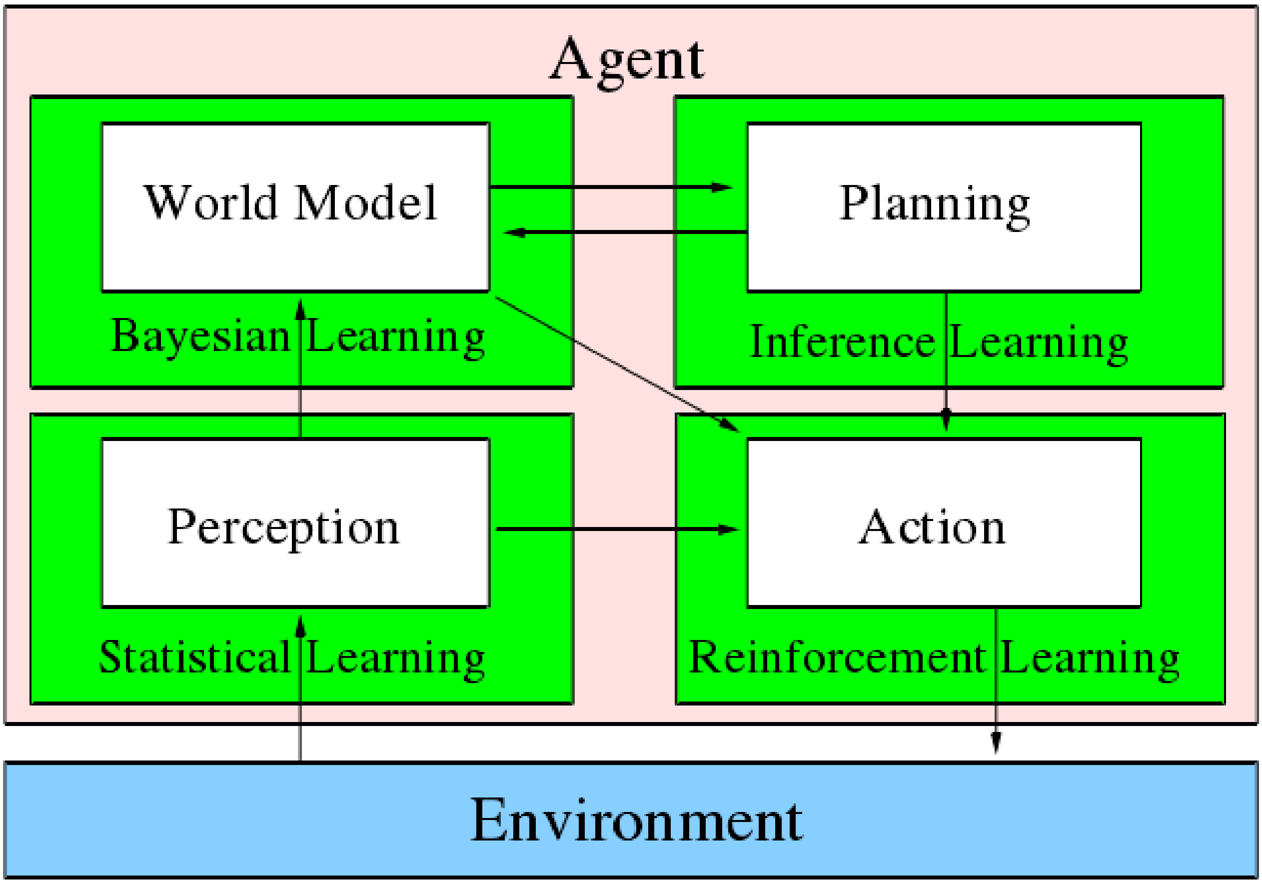
- 懂得分辨Predicates symbol 和 Function symbol

- ∃x∀y Loves(x, y)
  "There is a person who loves everyone in the world"

- ∀y∃x Loves(x, y)
  "Everyone in the world is loved by at least one person"

week9 Learning and Decison Tree

9.1 Learning Agent(必考)



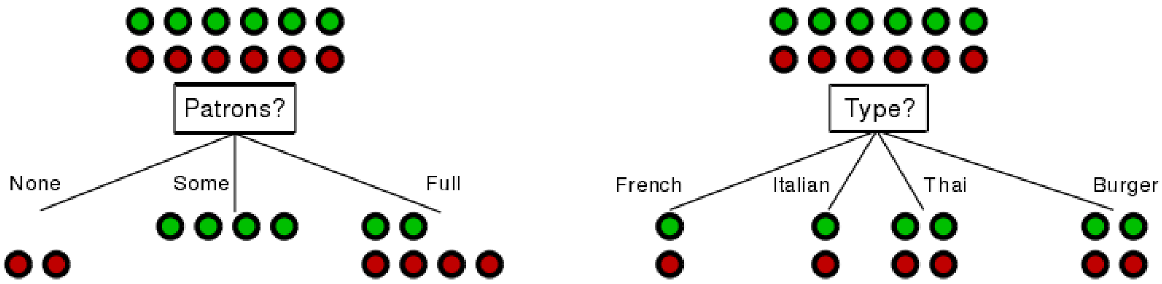
9.2 Learning Types

- 监督学习Supervised Learning
  agent is presented with examples of inputs and target outputs
  - 决策树、神经网络、SVM
    分为以下几个研究方向
  - framework (desion tree,NN,SVM ,etc)
  - representation (of input and output *归一化应该是*)
  - pre-processing / post-processing
  - training method(perceptron learning,BP,etc)
  - generalization (avoid over-fitting)
  - evaluation (separate training and test sets)
- 无监督学习
  agent is only presented with examples of inputs, aims to find structure
  - cluster, PCA, deep learning

- 增强学习 Reinforcement Learning
  agent is not presented with target outputs,but is given a reward signal,aims to maximize
- **奥卡姆剃刀**: avoid overfitting

9.3 熵 Entropy

- 因为训练数据有noise，使之不一致，为了确保产生正确的树，引入熵



$$\begin{aligned}\text{For Patrons, Entropy} &= \frac{1}{6}(0)+\frac{1}{3}(0)+\frac{1}{2}[-\frac{1}{3}\log(\frac{1}{3})-\frac{2}{3}\log(\frac{2}{3})] \\ &= 0+0+\frac{1}{2}[\frac{1}{3}(1.585)+\frac{2}{3}(0.585)] = 0.459 \\ \text{For Type, Entropy} &= \frac{1}{6}(1)+\frac{1}{6}(1)+\frac{1}{3}(1)+\frac{1}{3}(1) = 1\end{aligned}$$

- 图中Patrons attring比type要好，因为其分类的三个set 中有两个是完全集
- 熵越大代表信息越混乱，所以E越小越好
- Function: $\sum_i \frac{p_i+n_i}{p+n} H(< \frac{p_i}{p_i+n_i}, \frac{n_i}{p_i+n_i} >)$
- $H(< p_1, \ldots, p_n >) = \sum -p_i log_2 pi$
  suppose we have p positive and n nagative example at a node

9.4 Induce Tree (Pruning剪枝 Laplace Error)

- 根据奥卡姆剃刀, 我们需要对那些对分类增益不大的分支进行修剪
- $E = 1 - \frac{n+1}{N+K}$
- N = 总结点数
- n = number of items in the majority class

- K = number of classes
- 如果子节点的平均拉普拉斯误差超过父节点->剪枝
- e.g.1

  Should the children of this node be pruned or not?

  Left child has class frequencies [3,2]

  $$E = 1 - \frac{n+1}{N+k} = 1 - \frac{3+1}{5+2} = 0.429$$

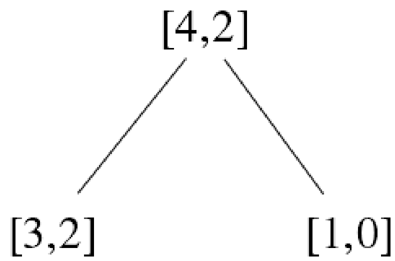
  Right child has $E = 0.333$

  Parent node has $E = 0.375$

  Average for Left and Right child is

  $$E = \frac{5}{6}(0.429) + \frac{1}{6}(0.333) = 0.413$$

  Since $0.413 > 0.375$, children should be pruned.



- e.g.2

  Should the children of this node be pruned or not?

  Left and Middle child have class frequencies [15,1]

  $$E = 1 - \frac{n+1}{N+k} = 1 - \frac{15+1}{16+2} = 0.111$$

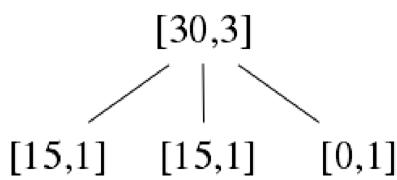
  Right child has $E = 0.333$

  Parent node has $E = \frac{4}{35} = 0.114$

  Average for Left, Middle and Right child is

  $$E = \frac{16}{33}(0.111) + \frac{16}{33}(0.111) + \frac{1}{33}(0.333) = 0.118$$

  Since $0.118 > 0.114$, children should be pruned.



9.5 Summary

- Supervised Learning
- Ockham's Razor: tradeoff between simplicity and accuracy
- Decision Trees
  - Generalisation: build smaller tree
  - Pruning: Laplace error

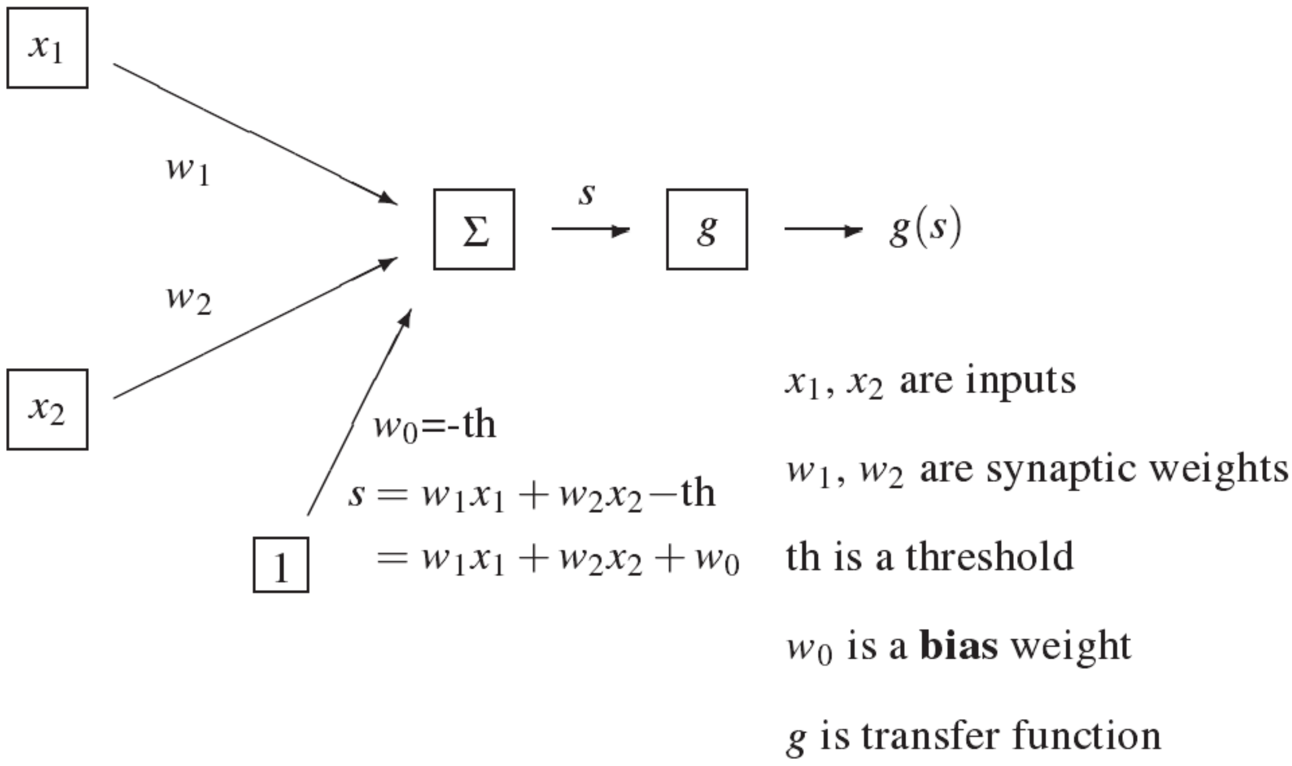
week10 Perceptrons and Neural Networks

感知器和神经网络讲的都比较基础，拓展可以看COMP9417课件

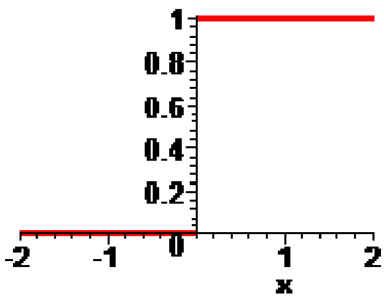
10.1 NN模型

- input edges
- output edges
- an activaion level(输入_激活函数)
  权重可以是积极的或消极的，并可能随着时间的推移改变（学习）。
  输入函数是activation level of inputs 的加权和。
  激活层是一个**非线性传递**函数
  $activaiton_i = g(s_i) = g(\sum_j w_{ij} x_j)$

10.2 感知器



- 采用的感知器为二元线性分类器，图为单层人工神经网络
- g 是转换函数（通常为sigmoid函数），这门课用二元分类函数



$$g(s) = \begin{cases} 1, & \text{if} \quad s \geq 0 \\ 0, & \text{if} \quad s < 0 \end{cases}$$

- Linear Separability 线性可分性

AND	$w_1 = w_2 = \quad 1.0,$	$w_0 = -1.5$
OR	$w_1 = w_2 = \quad 1.0,$	$w_0 = -0.5$
NOR	$w_1 = w_2 = -1.0,$	$w_0 = \quad 0.5$

- 学习规则，改变权重

Adjust the weights as each input is presented.

recall: $s = w_1 x_1 + w_2 x_2 + w_0$

if $g(s) = 0$ but should be 1,

$$
\begin{aligned}
w_k &\leftarrow w_k + \eta\, x_k \\
w_0 &\leftarrow w_0 + \eta \\
\text{so} \quad s &\leftarrow s + \eta\,(1 + \sum_k x_k^2)
\end{aligned}
$$

if $g(s) = 1$ but should be 0,

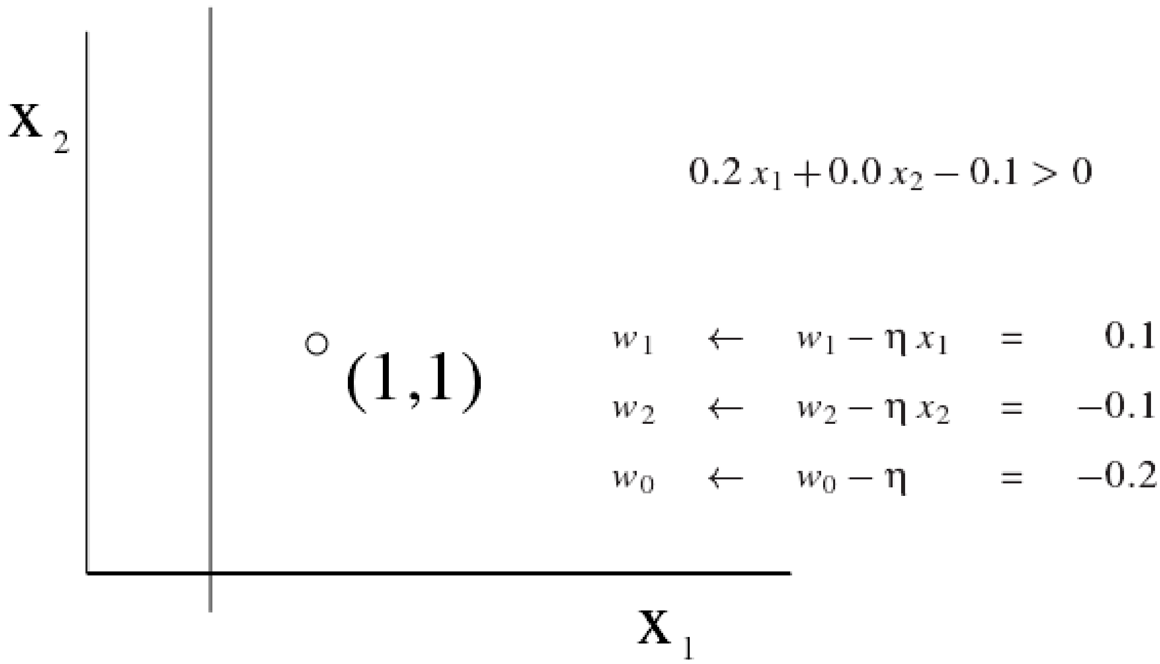
$$
\begin{aligned}
w_k &\leftarrow w_k - \eta\, x_k \\
w_0 &\leftarrow w_0 - \eta \\
\text{so} \quad s &\leftarrow s - \eta\,(1 + \sum_k x_k^2)
\end{aligned}
$$

otherwise, weights are unchanged. ($\eta > 0$ is called the **learning rate**)

Theorem: This will eventually learn to classify the data correctly, as long as they are **linearly separable**.

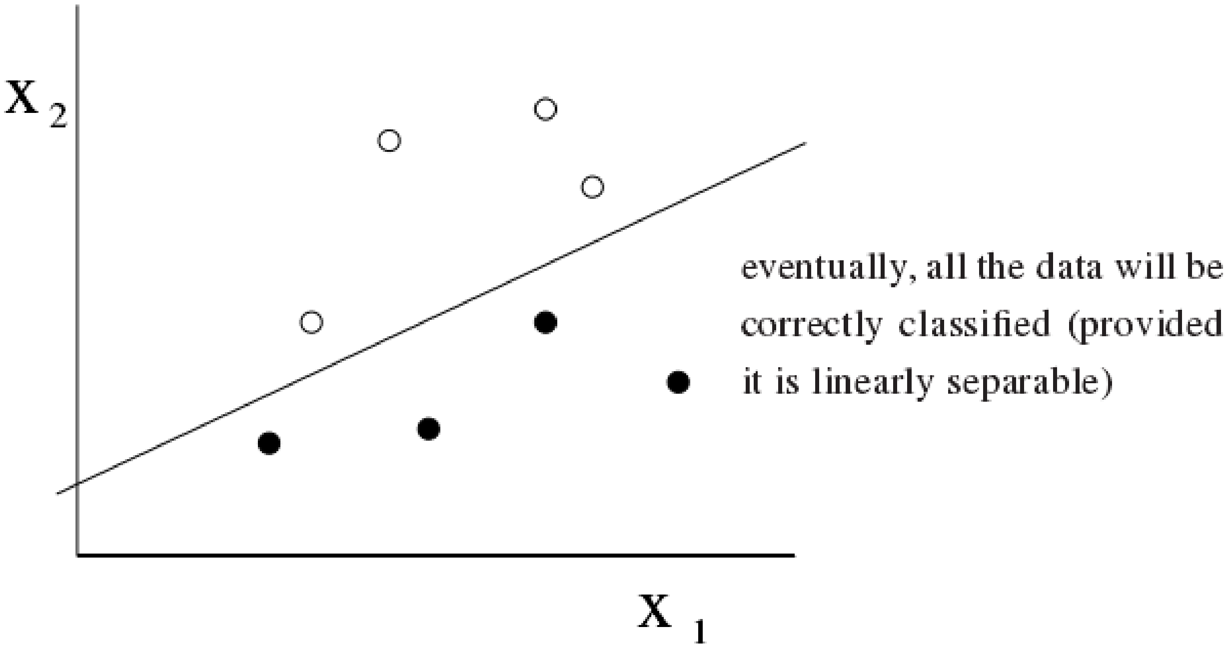
η 通常为0.1或1

- 举个例子
  - 判断函数 $w_1x_1 + w_2x_2 + w_0 > 0$
  - 学习速率 η =0.1
  - 初始化随机权重 $w_1 = 0.2, w_2 = 0.1, w_0 = -0.1$



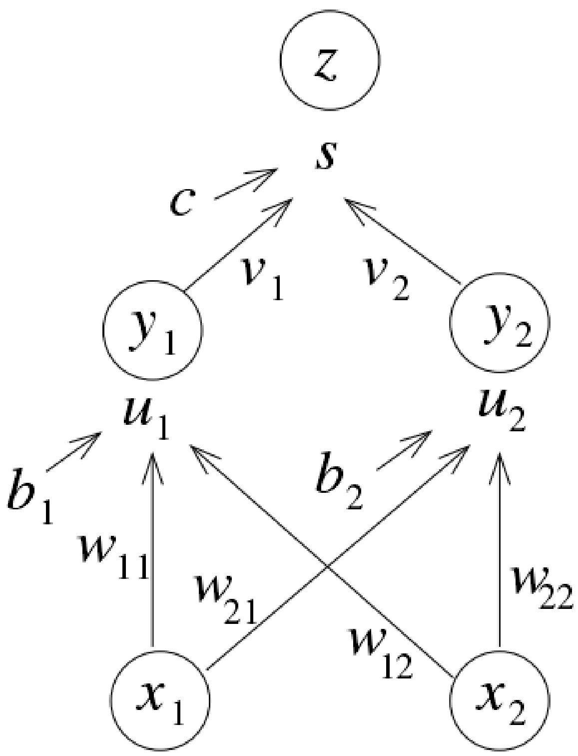
图中(1,1)不为实心，代表为负(判断函数为负)，但是将(1,1)代入函数为正，所以修改权重

- 最终所有结果被正确分类（*最后的权重为图中的分界线*）



10.3 Multi—Layer Nerual Netwrok

- 解决XOR异或问题
- 首先将感知器中的二元分类函数变为连续函数，如sigmoid函数
- variance方差，cost function(erro function)= $\frac{1}{2}\sum(y_i - \hat{y}_i)^2$
- 梯度下降，获得最小的E



$$
\begin{aligned}
u_1 &= b_1 + w_{11}x_1 + w_{12}x_2 \\
y_1 &= g(u_1) \\
s &= c + v_1 y_1 + v_2 y_2 \\
z &= g(s) \\
E &= \frac{1}{2}\sum(z-t)^2
\end{aligned}
$$

- BP算法不考*（有兴趣可以去看看，在之后AI的相关课程里很重要）*

10.4 training tips

- 输入数据和输出结果都需要归一化
- 权重随机初始化very small 的值 （eta为0.1，权重初始值太多会很耗时）
- online learning / batch learning
- 防止过拟合
  - 简化模型
  - 交叉检验 cross validation
  - weight decay
- 在梯度下载的过程中可调整η值

week11 Uncertainty

11.1 概念

- s

概率	描述
先验概率	事件发生前的预判概率。可以是基于历史数据的统计，可以由背景常识得出，也可以 是人的主观观点给出。一般都是单独事件概率，如P(x),P(y)
后验概率	事件发生后求的反向条件概率, 或者说，基于先验概率求得的反向条件概率。概率形式与条件概率相同
条件概率	一个事件发生后另一个事件发生的概率。一般的形式为P(x
似然概率	通过历史数据统计得到的条件概率, 一般不把它叫做先验概率，但从定义上也符合先验定义

- $$P(Cause|Effect) = \frac{P(Effect|Cause)P(Cause)}{P(Effect)}$$
  - P(Cause|Effect)为后验概率, 即为求解目标

- P(Effect|Cause)为似然概率
- P(Cause)为先验概率
- P(Effect) 其实也是先验概率，只是在贝叶斯的很多应用中不重要(因为只要最大后验不求绝 对值)，需要时往往用全概率公式计算得到

- 最大似然理论: 令似然函数P(x|y)最大的y'即为y的最大似然估计

$$Max(P(x|y)) = \prod_{k=1}^{n} P(x_k|y) \text{ , for all y}$$

-
  - 贝叶斯理论: 在利用总体信息和样本信息的同时, 还利用先验概率p(y)
    - 因为有可能某个y是很稀有的类别几千年才看见一次，即使P(x|y)很高，也很可能不是它
    - 贝叶斯理论存在的问题: 实践中先验概率可能并不准确

$$y = Max(P(x|y)P(y))$$

11.2 Inference

Start with the joint distribution:

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	**.108**	**.012**	**.072**	**.008**
¬ cavity	**.016**	**.064**	**.144**	**.576**

For any proposition ϕ, sum the atomic events where it is true:

$P(\phi) = \sum_{\omega:\omega\models\phi} P(\omega)$

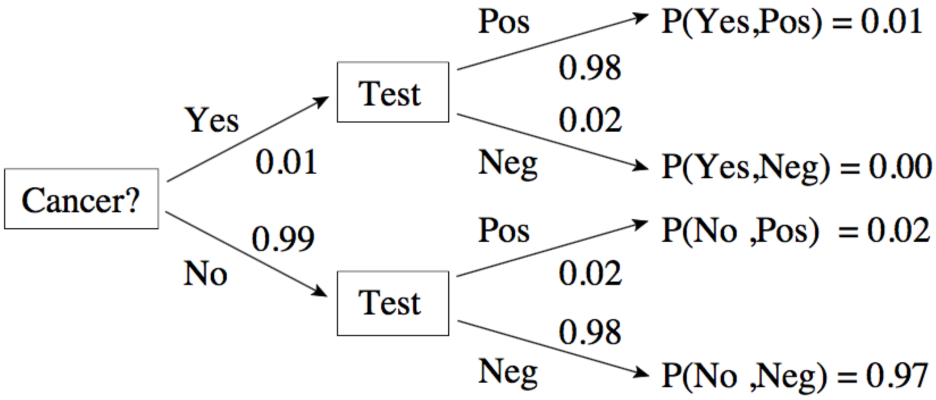
$P(toothache) = 0.108 + 0.012 + 0.016 + 0.064 = 0.2$

	toothache		*¬ toothache*	
	catch	*¬ catch*	*catch*	*¬ catch*
cavity	**.108**	**.012**	**.072**	**.008**
¬ cavity	**.016**	**.064**	**.144**	**.576**

Can also compute conditional probabilities:

$$P(\neg cavity | toothache) = \frac{P(\neg \texttt{cavity} \land \texttt{toothache})}{P(\texttt{toothache})}$$
$$= \frac{0.016 + 0.064}{0.108 + 0.012 + 0.016 + 0.064} = 0.4$$

11.3 Bayes'　Rule and Conditional Independence

- 

$$
\begin{aligned}
P(\texttt{cancer}\,|\,\texttt{positive}) &= \frac{P(\texttt{positive}\,|\,\texttt{cancer})P(\texttt{cancer})}{P(\texttt{positive})} \\
&= \frac{0.98*0.01}{0.98*0.01+0.2*0.99} = \frac{0.01}{0.01+0.02} = \frac{1}{3}
\end{aligned}
$$

week12 Learning Games (不考)